МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине

‘Системы искусственного интеллекта‘

Выполнил:
Студент группы P33312
Соболев Иван
Александрович
Преподаватель:
Кугаевских Александр
Владимирович

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2023

Задание:

- Получите и визуализируйте статистику по датасету (включая количество, среднее значение, стандартное отклонение, минимум, максимум и различные квантили).
- Проведите предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и нормировка.
- Разделите данные на обучающий и тестовый наборы данных.
- Реализуйте линейную регрессию с использованием метода наименьших квадратов без использования сторонних библиотек, кроме NumPy и Pandas. Использовать минимизацию суммы квадратов разностей между фактическими и предсказанными значениями для нахождения оптимальных коэффициентов.
- Постройте **три модели** с различными наборами признаков.
- Для каждой модели проведите оценку производительности, используя метрику коэффициент детерминации, чтобы измерить, насколько хорошо модель соответствует данным.
- Сравните результаты трех моделей и сделайте выводы о том, какие признаки работают лучше всего для каждой модели.
- Бонусное задание
    - Ввести синтетический признак при построении модели

Этапы реализации и пояснения:

На первых шагах были импортированы нужные библиотеки, далее с помощью библиотеки pandas считан датасет и выведена основная статистика по нему.

Дальше необходимо было сделать предварительную обработку и нормировку данных, для этого была написана функция нормирования.

```
def standardize_data(data):
    mean = np.mean(data, axis=0)
    std = np.std(data, axis=0)
    standardized_data = (data - mean) / std
    return standardized_data
```

Создаем функцию standardize_data, которая нормирует данные, приводя их к единому масштабу. Сначала мы вычисляем среднее значение (mean) и стандартное отклонение (std) для каждого признака. Затем мы вычитаем среднее значение из каждой точки данных и делим на стандартное отклонение, чтобы получить нормированные данные.

Когда данные прошли предварительную обработку их было необходимо разделить на тренировочные и тестовые.

```
# Определяем зависимую переменную (целевую) и признаки
X = data.drop(columns=['Performance Index'])
y = data['Performance Index']


# Разделение на обучающий и тестовый наборы
def train_test_split_custom(X, y, test_size=0.2):
    num_samples = X.shape[0]
```

```
    num_test_samples = int(test_size * num_samples)

    # Генерация случайных индексов для тестового набора
    test_indices = np.random.choice(num_samples, num_test_samples,
replace=False)

    # Индексы для обучающего набора
    train_indices = np.setdiff1d(np.arange(num_samples), test_indices)

    X_train, X_test = X.iloc[train_indices], X.iloc[test_indices]
    y_train, y_test = y.iloc[train_indices], y.iloc[test_indices]

    return X_train, X_test, y_train, y_test


X_train, X_test, y_train, y_test = train_test_split_custom(X, y,
test_size=0.2)
```

Создаем функцию train_test_split_custom, которая разделяет данные на обучающий и тестовый наборы. Эта функция случайным образом выбирает индексы для тестового набора данных, исходя из заданного коэффициента test_size. Таким образом, мы получаем два набора данных: X_train, y_train - обучающий набор, и X_test, y_test - тестовый набор.

Дальше создаём основной модуль линейной регрессии.

```
def perform_linear_regression(columns, X_train, X_test, y_train, y_test):
    # Добавляем столбец с единицами для учета свободного члена
    X_train['intercept'] = 1
    X_test['intercept'] = 1
    if columns != None:
        columns = columns.split(",")

        if len(columns) == 1:
            X_train = X_train[['intercept', columns[0]]]
            X_test = X_test[['intercept', columns[0]]]
        elif len(columns) == 2:
            X_train = X_train[['intercept', columns[0], columns[1]]]
            X_test = X_test[['intercept', columns[0], columns[1]]]
        else:
            X_train = X_train[['intercept', columns[0], columns[1],
columns[2]]]
            X_test = X_test[['intercept', columns[0], columns[1],
columns[2]]]
    # Преобразуем данные в массивы NumPy
    X_train = X_train.to_numpy()
    X_test = X_test.to_numpy()
    y_train = y_train.to_numpy()
    y_test = y_test.to_numpy()

    # Вычислим коэффициентов линейной регрессии методом наименьших квадратов
    coefficients = np.linalg.lstsq(X_train, y_train, rcond=None)[0]

    # Вычислим суммы квадратов
    def sum_of_squares(y_true, y_pred):
        return np.sum(np.square(y_true - y_pred))

    # Получим предсказания для тестового набора данных
    y_pred = np.dot(X_test, coefficients)

    # Оценка производительности с использованием коэффициента детерминации
(R^2)
    def r2_score_custom(y_true, y_pred):
        total_variance = np.sum((y_true - np.mean(y_true)) ** 2)
        residual_variance = np.sum((y_true - y_pred) ** 2)
```

```
        r2 = 1 - (residual_variance / total_variance)
        return r2

    # Вычислим R^2 для модели
    r2 = r2_score_custom(y_test, y_pred)
    sum_of_squares = sum_of_squares(y_test, y_pred)

    return y_pred, r2, sum_of_squares
```

В первой строчке добавляем свободный член, чтобы итоговая прямая не была привязана к началу координат. Далее преобразуем данные в массивы NumPy и затем используем функцию np.linalg.lstsq() для вычисления коэффициентов линейной регрессии методом наименьших квадратов. Результатом является массив coefficients, содержащий коэффициенты регрессии. После этого мы используем вычисленные коэффициенты для предсказания значений целевой переменной (y_pred) на тестовом наборе данных, умножая матрицу признаков X_test на вектор коэффициентов с помощью функции np.dot().

Далее просто создаем несколько моделей и анализируем их.

Выводы:

Можно заметить, что коэффициент детерминации сильно повысился за счёт Previous Scores и Hours Studied, можем сделать вывод, что успеваемость зависит от имеющихся знаний студента (предыдущих оценок), и также от того, сколько часов он потратил на учебу. Если у студента хорошие знания и мало учился, то скорее успеваемость у него будет немного меньше, а если хорошо подготовился, то будет примерно такой же балл. Если у студента плохие знания и много учился, то у него скорее всего сильно поднимется оценка. Также мотивация немного поднимает коэффициент детерминации, следовательно, чем выше мотивация, тем выше будет успеваемость.